## Estructura servicio Gestión de empleados:

## Constitución del Microservicio:

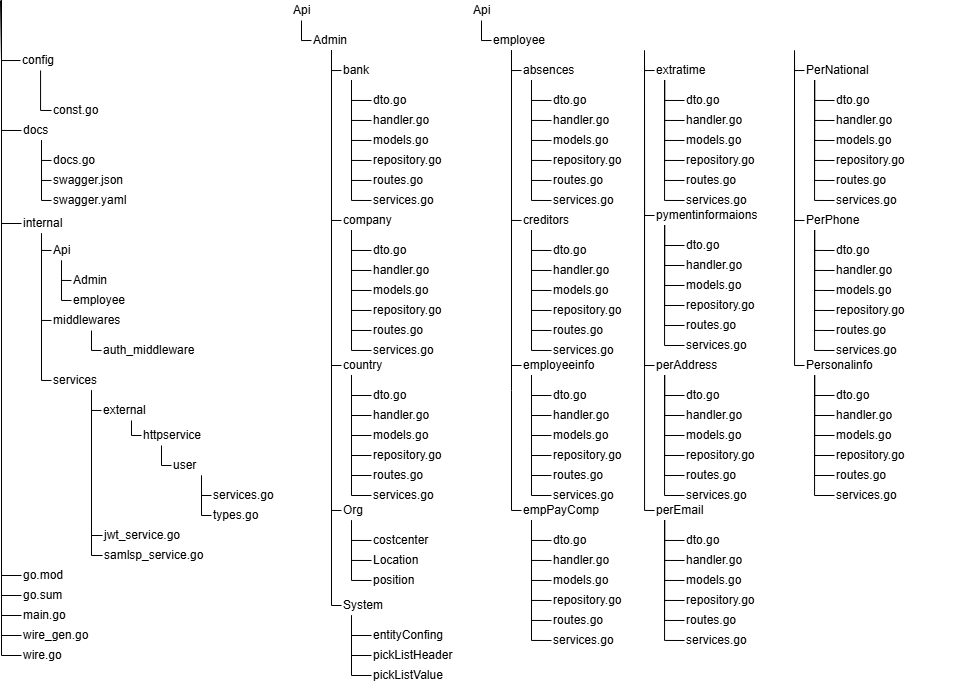
El aplicativo se encuentra desarrollado en el lenguaje de golang bajo la herramienta de Echo Framework.

Embebido a esto también se da uso de las siguientes librerías principales:

* **bitbucket.org/labs2022nbc/back-genericapi:**
* **fatih/structs:** Librería que permite transformar los strucks de go en maps con un índice como String y almacena una interfaz que pueden guardar cualquier tipo de dato.
* **google/wire :** Herramienta de inyección de dependencias para Go que ayuda a gestionar y organizar las dependencias en la aplicación.
* **joho/godotenv :** Herramienta para cargar variables de entorno desde un archivo .env, facilitando la gestión de configuraciones.
* **labstack/echo/v4 :** Framework web para Go, diseñado para crear aplicaciones y APIs de manera rápida y eficiente.
* **swaggo/echo-swagger :** Ayuda a documentar automáticamente las APIs y generar una interfaz visual para interactuar con ellas.
* **swaggo/swag :** Herramienta para generar documentación Swagger a partir del código Go, que facilita la creación de APIs bien documentadas.
* **mongo-driver:** Controlador oficial de MongoDB para Go, utilizado para interactuar con bases de datos MongoDB.

## Estructura del Microservicio:

* Se muestra el contenido de la carpeta raíz y el contenido de la carpeta Api-Admin:



* Se muestra el contenido de la carpeta Api-employee:

A diagram of a computer

Description automatically generated with medium confidence

**Carpeta internal:** La carpeta internal gestiona las solicitudes y la lógica de negocio, guarda los módulos de que utilizara la app, cada uno tiene una estructura similar.

* **Carpeta Api:**   En la carpeta Api se guarda la información los modelos y CRUDs que usara el sistema, en este caso solo **Admin** y **employee**, cada uno guarda sus atributos/tablas y mantiene la misma estructura de archivos **(dto.go, handler.go, routers.go, services.go, Repositorys.go).**
* **Carpeta admin:**
  + **Carpeta bank:**
* **dto.go:** Formato de los formularios y la estructura de los tipos de datos básicos del json
* **handler.go:** Maneja la solicitud en los endpoints
* **routers.go:** Aquí se manejan los endpoints y la lógica que manejara cada uno.
* **services:** Comunica la solicitud y la base de datos o el repositorio, mantiene la lógica de negocio.
* **Repositorys.go:** Se encarga la comunicación a la base de datos, guarda, elimina, envía los datos.
  + **Carpeta company:**
  + **Carpeta country:**
  + **Carpeta Org:**
  + **Carpeta System:**
* **Carpeta employee:**
  + **Carpeta absences**:
* **dto.go:** Formato de los formularios y la estructura de los tipos de datos básicos del json
* **handler.go:** Maneja la solicitud en los endpoints
* **routers.go:** Aquí se manejan los endpoints y la lógica que manejara cada uno.
* **services:** Comunica la solicitud y la base de datos o el repositorio, mantiene la lógica de negocio.
* **Repositorys.go:** Se encarga la comunicación a la base de datos, guarda, elimina, envía los datos.
  + **Carpeta creditor**:
  + **Carpeta employeeInfo**:
  + **Carpeta empPayComp**:
  + **Carpeta extraTime**:
  + **Carpeta paymentInformation**:
  + **Carpeta perAddress**:
  + **Carpeta perEmail**:
  + **Carpeta perNational**:
  + **Carpeta perPhone**:
  + **Carpeta personalInfo**:
  + **Carpeta services:**
  + **Carpeta externals:**
* **Carpeta httpservice:**
* **Service.go:** Este archivo contiene la lógica principal para el manejo de peticiones y respuestas en un servicio HTTP.
* **Types.go:** Este archivo define las estructuras (structs) y tipos de datos que el servicio HTTP necesita para funcionar.
* **Jwt\_service.go:** Ayuda a la creación de web tokens.
* **Samlsp\_services.go:** Ayuda permitir el servicio de samlsp

## Módulos:

* **Admin:**

El módulo **Admin** está diseñado para administrar diversas entidades clave del sistema, organizándolas en tablas específicas que representan distintos aspectos de la estructura administrativa. Cada tabla contiene datos y funcionalidades esenciales para la gestión eficiente de estas entidades, permitiendo realizar operaciones CRUD (crear, leer, actualizar y eliminar) de manera centralizada.

* **bank**: Información relacionada con las entidades bancarias utilizadas en el sistema.
* **company**: Nombre de la compañía que pertence.
* **country**: Información sobre los países.
* **org**: Datos relacionados con la organización, como la localización, la posición dentro de la empresa y centro de costo.
* **system**: Configuraciones específicos del sistema, para esto se genera una picklist.
* **Employee:**

El módulo **Employee** está diseñado para gestionar toda la información relevante sobre los empleados, organizando cada aspecto en tablas específicas, donde cada tabla representa un atributo relacionado con el empleado. Esto permite almacenar y manejar detalles particulares de los empleados de manera modular y estructurada.

* **absences**: Información sobre ausencias del empleado, registrando datos como fechas y razones.
* **creditor**: Detalles sobre los acreedores relacionados con el empleado.
* **employeeInfo**: Información general del empleado.
* **empPayComp**: Componentes de pago del empleado.
* **extraTime**: Registro de horas extra trabajadas por el empleado.
* **paymentInformation**: Información sobre los métodos de pago asociados al empleado.
* **perAddress**: Información de dirección personal del empleado.
* **perEmail**: Dirección de correo electrónico personal del empleado.
* **perNational**: Información sobre la nacionalidad y documentos de identidad.
* **perPhone**: Número de teléfono personal del empleado.
* **personalInfo**: Información personal sobre el empleado, complementa con el perpAddress, perEmail.PerNational y PerPhone.

## Variables de entorno:

* **DATABASE\_HEADER\_NAME:** Nombre del header que contiene la llave de la base de datos o del cluster.
* **MS\_CLUSTER\_ENDPOINT:** El endpoint del microservicio del cluster, se representa con una URL refiriendo en donde se encuentra el microservicio.
* **DATABASE\_KEY\_HEADER\_NAME:** llave del header que contiene la llave de la base de datos o del cluster.
* **USER\_ID\_HEADER\_NAME:** Nombre del header que contiene la llave del User ID.
* **SERVER\_PORT:** Puerto del servidor donde aceptara las peticiones.
* **MONGO\_HOST:** Dirección del servidor de MongoDB, almacena una url para conectar al servidor.
* **MONGO\_APP\_NAME:** Nombre de la aplicación que se usará para identificar la aplicación en MongoDB.
* **AUTH\_KEY\_AWS\_ARN:** ARN de la clave utilizada en AWS para la autenticación de usuarios.
* **AUTH\_KEY\_REFRESH\_AWS\_ARN:** ARN de la clave usada en AWS para la renovación de tokens de autenticación
* **AUTH\_PUBLIC\_REFRESH\_KEY\_PATH:** Ruta al archivo que contiene la clave pública utilizada para verificar los tokens de renovación
* **JWT\_ISSUER:** Identificador del emisor de los tokens JWT, que indica qué sistema o entidad generó el token.
* **JWT\_AUDIENCE:** Identificador de la audiencia para la cual los tokens JWT son válidos, es decir, los destinatarios del token.
* **AWS\_PROFILE:** Nombre del perfil de AWS utilizado para acceder a los servicios de AWS, como se define en tu configuración local de AWS.
* **MONGO\_DATABASE:** Nombre de la base de datos MongoDB que la aplicación va a usar.
* **FRONTEND\_URL:** URL de donde se hará la autenticación con SSO.
* **SSO\_MAIL\_ATTR:** Atributo de correo utilizado para Single Sign-On (SSO), que indica qué campo de correo electrónico se usa en el sistema de autenticación.